Richard G Baldwin (512) 223-4758, baldwin@austin.cc.tx.us,
http://www2.austin.cc.tx.us/baldwin/

The AWT Package, Arranging Components in Containers, FlowLayout

Java Programming, Lecture Notes # 116, Revised 03/03/99.

Preface

Students in Prof. Baldwin's **Intermediate Java Programming** classes at ACC are responsible for knowing and understanding all of the material in this lesson.

JDK 1.1 was formally released on February 18, 1997. This lesson was originally written on March 7, 1997 using the software and documentation in the JDK 1.1 download package. It has been updated on various occasions since that time.

The sample programs in this lesson were confirmed to compile and run properly under JDK 1.2 on 2/3/99.

Introduction

This is one in a series of lessons that concentrate on the package **java.awt** where most of the functionality exists for providing the user interface to your application or applet.

FlowLayout Manager

In earlier lessons, we have learned about absolute positioning and sizing of components in a container and have learned about the **BorderLayout** manager.

Absolute positioning allows you to place any (reasonable) number of components in a container, but has some problems with cross-platform compatibility.

The **BorderLayout** manager solves some of the cross-platform compatibility problems, but limits the number of components that can be placed in a container to a maximum of five. In addition, the placement of the components may or may not be suitable for your purposes.

JDK 1.1 provides the **FlowLayout** manager, which also solves the cross-platform compatibility problems and allows you to place any (reasonable) number of components in a container. In addition, the placement of the components may (or may not) be more suitable for your needs than the **BorderLayout**.

FlowLayout arranges components left to right until no more components fit on the same line. When no more components will fit on a line, a new line is started below the current line. Each line of components can either be *centered*, *left aligned*, or *right aligned* in the container.

If the container is resized by the user at runtime, the locations of the components will be automatically adjusted to place the maximum possible number of components on the first and each successive line.

The **FlowLayout** class has three *Fields*, three *Constructors*, and about a dozen *methods*.

In most cases, you will probably only be concerned with the *Fields* and the *Constructors* unless you need to dynamically modify the layout at runtime.

The three fields are:

- **CENTER --** The center alignment variable.
- **LEFT --** The left alignment variable.
- **RIGHT --** The right alignment variable.

The three constructors are:

- **FlowLayout()** -- Constructs a new Flow Layout with a centered alignment and a default five-unit horizontal and vertical gap.
- **FlowLayout(int)** -- Constructs a new Flow Layout with the specified alignment and a default five-unit horizontal and vertical gap.
- **FlowLayout(int, int, int)** -- Constructs a new Flow Layout with the specified alignment and the specified horizontal and vertical gaps. .

This lesson contains two sample programs. The first, a "plain vanilla" program, will use the third form of the constructor along with the **LEFT** alignment variable to construct a **FlowLayout** object with the components *left-aligned*, a ten-pixel horizontal gap between components, and a fifteen-pixel vertical gap between components.

The second sample program will make use of one of the methods to adjust the layout dynamically at runtime.

Note that as of 3/7/97, there are some typographical errors in the JDK 1.1 documentation package regarding the **FlowLayout** class. In particular, the documentation indicates that the class can only be used to layout buttons when in fact, it can be used for any component that is a subclass of **Component**.

Second, the documentation indicates that the Field **CENTER** corresponds to the **right** alignment variable instead of the **center** alignment variable as it should.

Plain Vanilla Sample Program

The following application, named Layout04, creates a "Plain Vanilla" visual object by placing five non-functional **Button** objects on a **Frame** object using the **FlowLayout** manager. (The buttons are non-functional because no event listener objects were created and registered on the buttons.)

Discussion of First Program

The **FlowLayout** object is constructed with LEFT alignment, a ten-pixel horizontal gap between components, and a fifteen-pixel vertical gap between components. The alignment and the gaps were defined by using the version of the constructor that allows for specification of these parameters. It is also possible to modify these parameters at runtime using methods of the class **FlowLayout**.

Manually resizing the **Frame** object at runtime causes the positions of the components to adjust automatically, placing the maximum possible number of components on the first line.

For simplicity, no event listeners were created and registered. Therefore, the "close" box is not operational and you will need to use some other method to terminate the program.

The program was tested using JDK 1.1 running under Win95.

Interesting Code Fragments in First Program

There isn't much code to be discussed in this program. The first statement creates a new **Frame** object. The second statement sets the layout manager for the **Frame** object to be **FlowLayout** with **LEFT** alignment, a horizontal gap of 10 pixels between components, and a vertical gap of 15 pixels between components.

The third statement is typical of the statements used to **add** the components to the **Frame** object.

```
    Frame myFrame = new Frame(
```

```
                              "Copyright 1997,
R.G.Baldwin");
    myFrame.setLayout(//align,Hgap,Vgap
                      new
FlowLayout(FlowLayout.LEFT,10,15));
    myFrame.add(new Button("First"));
```

A complete listing of the program is provided in the next section.

Program Listing of First Program

This section contains a complete listing of the program.

```
/*File Layout04.java Copyright 1997, R.G.Baldwin
Revised 10/28/97 to better accommodate the width
requirements of the document.

This program is designed to be compiled and run under
JDK 1.1

This program creates a "Plain Vanilla" visual object by
placing five non-functional Button objects on a Frame
object using the FlowLayout manager. The buttons are
non-functional because no event listener objects were
created and registered on the buttons.

The FlowLayout object is constructed with LEFT alignment,
a ten-pixel horizontal gap between components, and a
fifteen-pixel vertical gap between components.

Resizing the Frame object causes the positions of the
components to adjust automatically.

For simplicity, no event listeners were created and
registered.  Therefore, the "close" box is not operational
and you will need to use some other method to terminate
the program.

The program was tested using JDK 1.1.3 running under Win95.
*/
//=======================================================//

import java.awt.*;
import java.awt.event.*;
//=======================================================//
public class Layout04 {
  public static void main(String[] args){
    //instantiate a Graphical User Interface object
    GUI gui = new GUI();
  }//end main
}//end class Layout04
//=======================================================//
```

```
class GUI {
  public GUI(){//constructor
    Frame myFrame = new Frame(
                            "Copyright 1997, R.G.Baldwin");
    myFrame.setLayout(//align,Hgap,Vgap
                      new FlowLayout(FlowLayout.LEFT,10,15));
    myFrame.add(new Button("First"));
    myFrame.add(new Button("Second"));
    myFrame.add(new Button("Third"));
    myFrame.add(new Button("Fourth"));
    myFrame.add(new Button("Fifth"));
    myFrame.setSize(250,150);
    myFrame.setVisible(true);
  }//end constructor
}//end class GUI definition
//=========================================================//
```

Second Sample Program

This program has more substance that the previous one. Although it doesn't do anything particularly useful, it does illustrate how to modify a layout dynamically at runtime.

Discussion

This program is designed to be compiled and run under JDK 1.1

Five buttons are added to a frame using a **FlowLayout** object as the layout manager with a three-pixel gap between components in both the horizontal and vertical direction.

An **ActionListener** object is instantiated and registered to listen for action events on all five of the buttons, with all five buttons sharing the same event handler.

The behavior of the **ActionEvent** handler is to increase the spacing between components whenever any of the buttons is pressed. This is accomplished by

- increasing the **Vgap** and **Hgap** attributes of the **FlowLayout** object,
- setting the layout manager of the frame to be the modified **FlowLayout** object, and
- *validating* the frame.

The validation step is required in order for the change to take effect and become visible.

Repeatedly clicking any of the buttons causes the buttons to move apart. Eventually they will not all fit on the frame and they begin to disappear off the edges. They can be made to reappear by enlarging the frame.

A **windowClosing()** event listener object is instantiated and registered on the frame to terminate the program when the frame is closed.

The program was tested using JDK 1.1 running under Win95.

Interesting Code Fragments

This program contains several interesting code fragments.

The following code fragment instantiates a **FlowLayout** object for **CENTER** alignment with a three-pixel gap between components in both the horizontal and vertical directions. This object is then passed to the **setLayout()** method to establish the layout manager for the **Frame** object.

```
    FlowLayout myFlowLayout =
                        new
FlowLayout(FlowLayout.CENTER,3,3);
    myFrame.setLayout(myFlowLayout);
```

The following code is typical of that used to instantiate an **ActionListener** object and register it on all five buttons.

```
    MyActionListener myActionListener =
        new
MyActionListener(myFlowLayout,myFrame);
    button1.addActionListener(myActionListener);
```

The following is the operative code in the **ActionListener** object that modifies the layout dynamically at runtime.

This code responds whenever one of the buttons is clicked, using the **get** and **set** methods on the **Vgap** and **Hgap** attributes of the **FlowLayout** object to modify the spacing between components in the layout object.

Then the **setLayout()** method is used to establish the modified layout object as the layout manager for the **Frame** object. After that, the **Frame** object is *validated*, a process that is necessary to cause the changes to become effective and to become visible.

```
  public void actionPerformed(ActionEvent e){
    myFlowLayoutObject.setHgap(
                        myFlowLayoutObject.getHgap()
+ 5 );
    myFlowLayoutObject.setVgap(
                        myFlowLayoutObject.getVgap()
+ 5 );
    myFrameObject.setLayout(myFlowLayoutObject);
    myFrameObject.validate();
  }//end actionPerformed()
```

Program Listing

A complete listing of the program follows.

```
/*File Layout05.java Copyright 1997, R.G.Baldwin
Revised 10/28/97 to better accommodate the display width
of the document.

This program is designed to be compiled and run under
JDK 1.1

This program has more substance that the previous one in
this lesson. Although it doesn't do anything particularly
useful, it does illustrate how to modify a flow layout
dynamically at runtime.

Five buttons are added to a frame using a FlowLayout object
as the layout manager with a three-pixel gap between
components in both the horizontal and vertical direction.

An action listener object is instantiated and registered to
listen for action events on all five of the buttons, with
all five buttons sharing the same event handler.

The behavior of the ActionEvent handler is to increase the
spacing between components whenever any of the buttons is
pressed.  This is accomplished by increasing the Vgap and
Hgap attributes of the FlowLayout object, setting the
layout manager of the frame to the modified FlowLayout
object, and validating the frame.  The validation step is
required in order for the change to become visible.

Repeatedly clicking any of the buttons causes the buttons
to move apart. Eventually they will not all fit on the
frame and they begin to disappear off the edges.  They can
be made to reappear by enlarging the frame.

A windowClosing() event listener object is instantiated and
registered on the frame to terminate the program when the
frame is closed.

The program was tested using JDK 1.1.3 running under Win95.
*/
//=======================================================//

import java.awt.*;
import java.awt.event.*;
//=======================================================//
public class Layout05 {
  public static void main(String[] args){
    //instantiate a Graphical User Interface object
    GUI gui = new GUI();
  }//end main
}//end class Layout05
```

```
//=====================================================//

class GUI {
  public GUI(){//constructor
    Frame myFrame = new Frame(
                             "Copyright 1997, R.G.Baldwin");
    //Instantiate a FlowLayout object with CENTER
    // alignment and a Vgap and Hgap of 3 pixels.
    FlowLayout myFlowLayout =
                     new FlowLayout(FlowLayout.CENTER,3,3);
    //Set the layout manager for the frame to be the
    // FlowLayout object.
    myFrame.setLayout(myFlowLayout);

    //Instantiate five Button objects
    Button button1 = new Button("First");
    Button button2 = new Button("Second");
    Button button3 = new Button("Third");
    Button button4 = new Button("Fourth");
    Button button5 = new Button("Fifth");

    //Add the five Button objects to the Frame object in
    // the order specified.
    myFrame.add(button1);
    myFrame.add(button2);
    myFrame.add(button3);
    myFrame.add(button4);
    myFrame.add(button5);

    myFrame.setSize(250,150);
    myFrame.setVisible(true);

    //Instantiate an action listener object and register it
    // on all five buttons.
    MyActionListener myActionListener =
        new MyActionListener(myFlowLayout,myFrame);
    button1.addActionListener(myActionListener);
    button2.addActionListener(myActionListener);
    button3.addActionListener(myActionListener);
    button4.addActionListener(myActionListener);
    button5.addActionListener(myActionListener);

    //Instantiate and register a window listener to
    // terminate the program when the Frame is closed.
    myFrame.addWindowListener(new Terminate());
  }//end constructor
}//end class GUI definition
//=====================================================//

class MyActionListener implements ActionListener{
  FlowLayout myFlowLayoutObject;
  Frame myFrameObject;

  //constructor
  MyActionListener(FlowLayout layoutObject,Frame inFrame){
    myFlowLayoutObject = layoutObject;
```

```
    myFrameObject = inFrame;
  }//end constructor

  //When an action event occurs, increase the horizontal
  // and vertical gap between components in the FlowLayout
  // object.  Then set the layout manager for the frame to
  // be the newly-modified FlowLayout object.  Then
  // validate the frame to ensure a valid layout so that
  // the new visual will take effect.
  public void actionPerformed(ActionEvent e){
    myFlowLayoutObject.setHgap(
                        myFlowLayoutObject.getHgap() + 5 );
    myFlowLayoutObject.setVgap(
                        myFlowLayoutObject.getVgap() + 5 );
    myFrameObject.setLayout(myFlowLayoutObject);
    myFrameObject.validate();
  }//end actionPerformed()
}//end class MyActionListener


//=======================================================//

class Terminate extends WindowAdapter{
  public void windowClosing(WindowEvent e){
    //terminate the program when the window is closed
    System.exit(0);
  }//end windowClosing
}//end class Terminate
//=======================================================//
```

Review

Q - Without viewing the solution that follows, write a Java application that starts with a Frame object on the screen with a width of 280 and a height of 100 pixels. Use the **setSize()** method on the Frame object to obtain this exact size.

Your name must appear in the banner at the top of the frame.

The frame contains five square buttons with no captions. The five buttons are in a row across the client area of the frame with equal separation between the buttons.

The buttons are separated from one another by a gap of approximately 3 pixels.

When you click on any of the buttons, the gap between the buttons increases by 5 pixels and the size of every button is decreased.

As you continue to click on the buttons, the gap continues to increase by five pixels each click, the buttons continue to decrease in size, and the horizontal position of each button stays the same. In other words, the horizontal distance between the centers of the buttons does not change as the size of the gap between them increases.

Eventually, the buttons become so small that they disappear.

When you close the frame, the program terminates.

A - See solution below.

```
/*File SampProg136.java Copyright 1997, R.G.Baldwin

The program starts with a Frame object on the screen with
a size of 280 wide by 100 pixels tall.

Your name must appear in the banner at the top of the
frame.

The frame contains five square buttons with no captions.
The five buttons are in a row across the client area
of the frame with equal separation between the buttons.

The buttons are separated from one another by a gap of
approximately 3 pixels.

When you click on any of the buttons, the gap between
the buttons increases by 5 pixels and the size of every
button is decreased.

As you continue to click on the buttons, the gap continues
to increase by five pixels each click, the buttons
continue to decrease in size, and the horizontal
position of each buttons stay the same.  In other words,
the horizontal distance between the centers of the
buttons does not change as the size of the gap between
them increases.

Eventually, the buttons become so small that they
disappear.

When you close the frame, the program terminates.

The program was tested using JDK 1.1.3 running under Win95.
*/
//=======================================================//

import java.awt.*;
import java.awt.event.*;
//=======================================================//
public class SampProg136 {
  public static void main(String[] args){
    //instantiate a Graphical User Interface object
    GUI gui = new GUI();
  }//end main
}//end class SampProg136
//=======================================================//

class GUI {
```

```
  public GUI(){//constructor
    Frame myFrame = new Frame(
                            "Copyright 1997, R.G.Baldwin");
    //Instantiate a FlowLayout object with CENTER
    // alignment and a Vgap and Hgap of 3 pixels.
    FlowLayout myFlowLayout =
                    new FlowLayout(FlowLayout.CENTER,3,3);
    //Set the layout manager for the frame to be the
    // FlowLayout object.
    myFrame.setLayout(myFlowLayout);

    //Instantiate five MyButtonClass objects
    MyButtonClass button1 = new MyButtonClass();
    MyButtonClass button2 = new MyButtonClass();
    MyButtonClass button3 = new MyButtonClass();
    MyButtonClass button4 = new MyButtonClass();
    MyButtonClass button5 = new MyButtonClass();

    //Add the five MyButtonClass objects to the Frame
    // object in the order specified.
    myFrame.add(button1);
    myFrame.add(button2);
    myFrame.add(button3);
    myFrame.add(button4);
    myFrame.add(button5);

    myFrame.setSize(280,100);
    myFrame.setVisible(true);

    //Instantiate an action listener object and register
    // it on all five buttons.
    MyActionListener myActionListener =
                new MyActionListener(myFlowLayout,myFrame);
    button1.addActionListener(myActionListener);
    button2.addActionListener(myActionListener);
    button3.addActionListener(myActionListener);
    button4.addActionListener(myActionListener);
    button5.addActionListener(myActionListener);

    //Instantiate and register a window listener to
    // terminate the program when the Frame is closed.
    myFrame.addWindowListener(new Terminate());
  }//end constructor
}//end class GUI definition
//=======================================================//

class MyActionListener implements ActionListener{
  FlowLayout myFlowLayoutObject;
  Frame myFrameObject;

  //constructor
  MyActionListener(FlowLayout layoutObject,Frame inFrame){
    myFlowLayoutObject = layoutObject;
    myFrameObject = inFrame;
  }//end constructor
```

```
  //When an action event occurs, increase the horizontal
  // and vertical gap between components in the FlowLayout
  // object.  Then set the layout manager for the frame to
  // be the newly-modified FlowLayout object.  Then
  // validate the frame to ensure a valid layout so that
  // the new visual will take effect.
  //Each time the frame is validated, the overridden
  // getPreferredSize() method is called to determine the
  // preferred size of the buttons.  This method reduces
  // the preferredSize by five pixels each time it is
  // called to offset the increase in the gap between the
  // buttons.  This, in turn, causes the buttons to remain
  // centered in the same horizontal position even though
  // the gap between them is increasing.

  public void actionPerformed(ActionEvent e){
    myFlowLayoutObject.setHgap(
                        myFlowLayoutObject.getHgap() + 5 );
    myFlowLayoutObject.setVgap(
                        myFlowLayoutObject.getVgap() + 5 );
    myFrameObject.setLayout(myFlowLayoutObject);
    myFrameObject.validate();
  }//end actionPerformed()
}//end class MyActionListener

//=======================================================//

//Extend the Button class to make it possible to override
// the getPreferredSize() method.  The overridden
// method decreases the size of the button by five pixels
// each time it is called.
class MyButtonClass extends Button{
  int side = 55;

  public synchronized Dimension getPreferredSize(){
    side -= 5;
    return new Dimension(side,side);
  }//end getPreferredSize()
}//end MyButtonClass

//=======================================================//

class Terminate extends WindowAdapter{
  public void windowClosing(WindowEvent e){
    //terminate the program when the window is closed
    System.exit(0);
  }//end windowClosing
}//end class Terminate
//=======================================================//
```

-end-